Computer Graphics

ANIMATED OBJECT

Goalkeeper Game Using Python Turtle Model

By.
Mohammed Salahadin
Abdulrahman Tawfiq
Shakar Abdulrahman

Introduction

Animation at the present time is one of the most important skills that most programmers aspire to master. As new programmers in the programming world, we are looking forward to acquiring this skill in an easy, fast, and short way. So, we decided to start with the Animated Object project. In this project there will be a goalkeeper to block the balls directed at him. There will be a win-lose outcome. In this case, we will reach the desired result in this project which is moving an object via the keyboard.

Background Research

We chose this idea because it is one of the most important ideas in terms of implementation. The second purpose is to learn new skills and develop them in the world of animation. In addition, we are looking forward to developing this game in the future, as this project will be the first step.

Methodology and Approach

We use the python programming language to create an animated object which can be moved using keyboard and mouse. We chose this programming language because it is worth the time, it's readable and maintainable code and multiple programming paradigms, compatible with major platforms and systems, robust standard library and many open source frameworks and tools and simplify complex software development and adopt test driven development, and easy to use including the many futures and package that supports drawing the objects and change it's directions like Turtle Package which we use.

We have chosen an open source cross platform integration development environment for scientific programming in the python language which is Spyder.

Implementation

First Steps to the implementation

1. We opened a window by using turtle library with the Goalkeeper title and a logo (which is included below).
2. The size of the game window is exactly the half size (width, height) of the screen.
3. We created the background of the game from scratch by using numpy and cv2 libraries. Then, we used the bgpic method related to the turtle library to set the new image as a background to the opened window. The size of the background image is exactly the same as the game window.

Steps for creating background image for the playground

1. Creating a black image by using numpy with the same size of the window. Giving it a channel that equal to 3 to let everything to be drawn on it to be coloured, Ch=3 means (R, G, B). Otherwise, we will only get the gray scale image. Screenshot of the game window, its title, logo, and the background image created (Figure-1):
2. Drawing a green rectangle over the black image with the same size.
3. Drawing another smaller rectangle with a white border over the image.
4. Drawing the centred circle according to the centre point of the window.
5. Drawing the rest with the same idea and by manipulating with the starting and ending points.
6. Saving the image by using imwrite() method in cv2 with a specific new path and extension.
7. Using the bgpic() method in turtle to set the new image in the background of the window.

(Png logo that we have used)

(Figure-1)

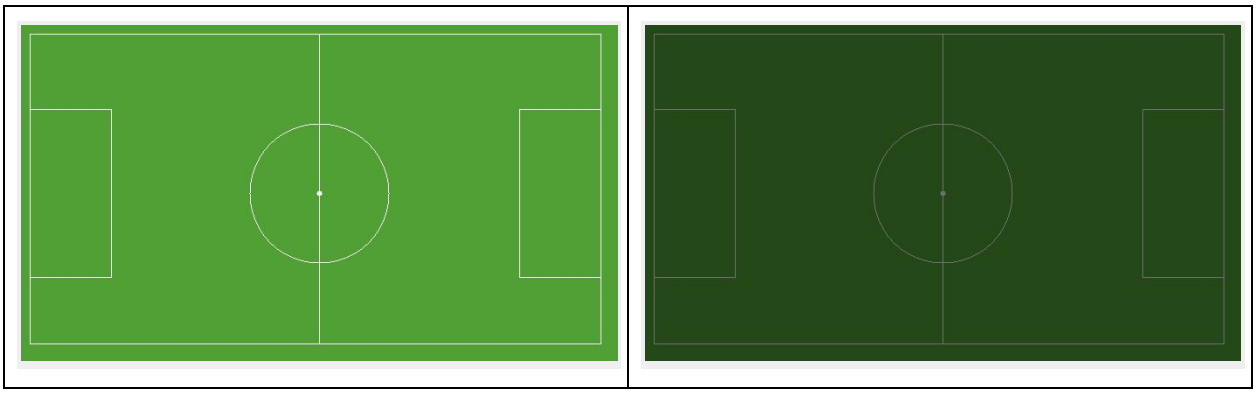
```
68
69  #Creating a black image in the background with the same size of the game window
70  #it is colored with ch=3
71  blackBackground=np.ones((gameScreenHeight,gameScreenWidth,3),dtype="uint8")
72
73
74
75
76  #We are drawing shapes over the black image that we created
77
78  #1- Green Rectangle is over the blackBackground
79  playGround=cv2.rectangle(blackBackground,(0,0),(gameScreenWidth,gameScreenHeight),(54,160,80),-1)
80
81  #2- Outer Border rectangle
82  playGroundOuterBorder=cv2.rectangle(playGround,(10,10),(gameScreenWidth-20,gameScreenHeight-20),(255,255,255),1)
83
84  #3- play Ground Middle Line
85  playGroundOuterMiddleLine=cv2.line(playGround,(halfGameScreenWidth,10),(halfGameScreenWidth,gameScreenHeight-20),(255,255,255),1)
86
87  #4- middle circle
88  middleCircle=cv2.circle(playGround,(halfGameScreenWidth,halfGameScreenHeight),80,(255,255,255),1)
89
90  #5- center point
91  centerPoint=cv2.circle(playGround,(halfGameScreenWidth,halfGameScreenHeight),3,(255,255,255),-1)
92
93  #6- penalty area 1
94  penaltyArea1=cv2.rectangle(playGround,(gameScreenWidth-thirdOfhalfGameScreenWidth,quarterGameScreenHeight),(gameScreenWidth-20,gameScreenHeight-quarterGameScreenHeight),(255,255,255),1)
95
96  #7- penalty area 2
97  penaltyArea2=cv2.rectangle(playGround,(10,quarterGameScreenHeight),(thirdOfhalfGameScreenWidth-10,gameScreenHeight-quarterGameScreenHeight),(255,255,255),1)
98
99
100
101
102
103
104 #saving the new background img that we created
105 cv2.imwrite("C:/Users/Ahmad/Desktop/background.png",blackBackground)
106 #put the new background img to our game window
107 window.bgpic("C:/Users/Ahmad/Desktop/background.png")
108
```

(Figure-2)

Game Modes

Enable night mode when user press number 2 on keyboard and return to normal mode when user presses 1. Buttons will be added later to control it as well.

Screenshots that shows the two modes



```
#Creating the night mood image and save it with backgroundNight name in images directory

RGB_Image2=cv2.imread("images/backgroundDay.png")

for r in range(RGB_Image2.shape[0]):

   for c in range(RGB_Image2.shape[1]):

      for ch in range(RGB_Image2.shape[2]):

         RGB_Image2[r][c][ch]=RGB_Image2[r][c][ch]*0.45 #decrease 55% of the value of each color R, G, B to make them colse to the 0 (black) to get the night mood

cv2.imwrite("images/backgroundNight.png",RGB_Image2)
```

Game Loading

A Loading page shows during the running time of the game until the main content of the game is ready. Adding a loading image that during the loading time of the program and disappears after starting the program:

Game Effects

Creating a code that shows the playground blurred in the beginning of starting the game then slowly zooming out and removing the blurring effects to show the playground. For the effect in the beginning of the game and after starting, we created a while loop to read the background image and change its properties in terms of blur and scaling. In each loop, the image will have different properties and will be saved in a specific name in a specific directory. We called the update method in turtle model to update the screen in each loop to show the changes in a live way. We put a try, except, finally inside the loop in order to read the images directly from the directory if the game will be opened for the next time. If the game opens for the first time, an exception will be accrued and the code of creating the images will be executed. We put the decrements and update inside the finally block as this block will be implemented in case of try or except. This was to decrease code duplication. So, as a final result, all these images will be readed inside the loop to get that effect.

```
#adding effects (bluring and scaling)
F_blur=50
F_scale=2
while F_blur>1:
    try:
        #if the app opened before, means all the picyures are in the dir and no need to create them again
        #this try to check the image if it is not found it will go to the exception and if not it gose to finally
        window.bgpic("images/bluring/background"+str(F_blur)+".png")
    except Exception as e:
        #if the error of file not found accures, this code is going to be executed to create all the images.

        #make the bg img blured
        blured=cv2.blur(RGB_Image, (F_blur,F_blur))
        #scaling it
        bluredScaled=cv2.resize(blured, None,fx=F_scale,fy=F_scale)
        #saving it with the new properties in new name and in the bluring directory
        cv2.imwrite("images/bluring/background"+str(F_blur)+".png",bluredScaled)
        #make it as bg in the window
        window.bgpic("images/bluring/background"+str(F_blur)+".png")
    finally:
        #this part is going to be excuted in all cases, so the update, increasing values, and sleep is here

        #decreasing the properties
        F_blur-=1
        F_scale-=0.02
        #updating the window to see the live changing
        window.update()
        #adding some delay after importing the time above
        time.sleep(0)
```

Animated painting

After closing the game by clicking on the close button a tkinter window will be opened and an animated word will be written on the screen, then the screen will close automatically after painting in a specific effect.

Game Objects

The goalkeeper object is added as a gif image. We have added the goalKeeper as a shape, and the turtle is hidden till the showturtle() method will be called to avoid the arrow to be shown. The speed that turtle module draws the shape on screen whenever it moves to let us see it. We have adjusted the speed of the Goalkeeper object.

```
# the speed that turtle module draws the shape on screen whenever it moves to let us see it
goalKeeper.speed(0) #to let the shape updated in the maximum speed
```

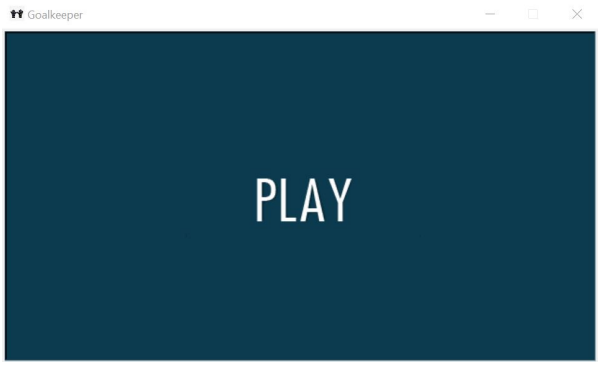
In the turtle model we have only these shapes ['arrow', 'blank', 'circle', 'classic', 'square', 'triangle', 'turtle']. But we can add a shape as a gif image using the addshape method. We use the shape method to get the shape on the window and we use our shape instead of the ready shapes like shape (“Square”) or shape (“circle”). show the shape on the screen when it is in the suitable position. The same thing for the ball and the machine launcher.(Add shape doesn’t accept any other type of extensions rather than gif extension)

```
#turtle model has these shapes ['arrow', 'blank', 'circle', 'classic', 'square', 'triangle', 'turtle']
#but we can add a shape as gif image using the addshape method
turtle.addshape(name="images/goalkeeper.gif",shape=None)
#we use the shape method to get the shape on the window and we use our shape instead of the ready shapes like shape("square
goalKeeper.shape("images/goalkeeper.gif")
```

Other Buttons “Shapes” Added:

1. Play shape “Clickable”, once the game starts “start() be called”, the play button will be disappeared
2. button to change the day mode
3. button to change the night mode
4. button to close the window
5. button to pause the game

Screenshots

	
Clickable Play Button	Buttons as Menu Above

Game Animation

Objects Go Inside The Playground

We added animation when the game starts to move the goalkeeper, ball, and the machine launcher from outside the playground to inside it by using a loop for the goalkeeper, ball, and the machine objects.

```
# x coordinates for them before the loop of the animation
loop=0
keeper_w=360
ball_w=-345
machine_w=-360
#animation loop for the goalkeeper, ball, and the machine objects
while loop<25 :
    goalKeeper.goto(keeper_w,0)
    ball.goto(ball_w,0)
    machine.goto(machine_w,10)
    keeper_w-=3.5
    ball_w+=8.5
    machine_w+=8.5
    loop+=1
    window.update()
```

loop animation for moving the objects to the playground

We have created functions to move the goalKeeper up and down by using the keyboard keys W for calling the function to change the position of goalKeeper to go to up, and another key which is S for calling the function to move his position to go down.

Methodology:

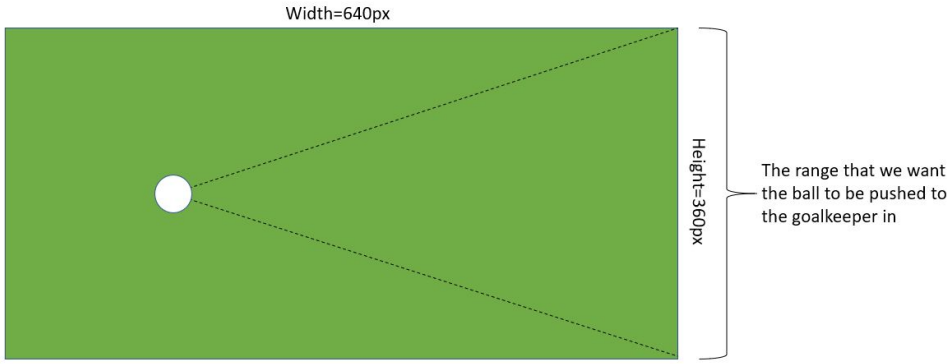
- Get the current y coordinate of the goalkeeper object
- Decrease current y coordinate by a specific pixels each time we move the goalkeeper down
- Set the new y coordinate

We have limited the goalkeeper from going outside the playground with each movement, each time he moves we check his position and compare it with the playground position in order to not go out of the playground.

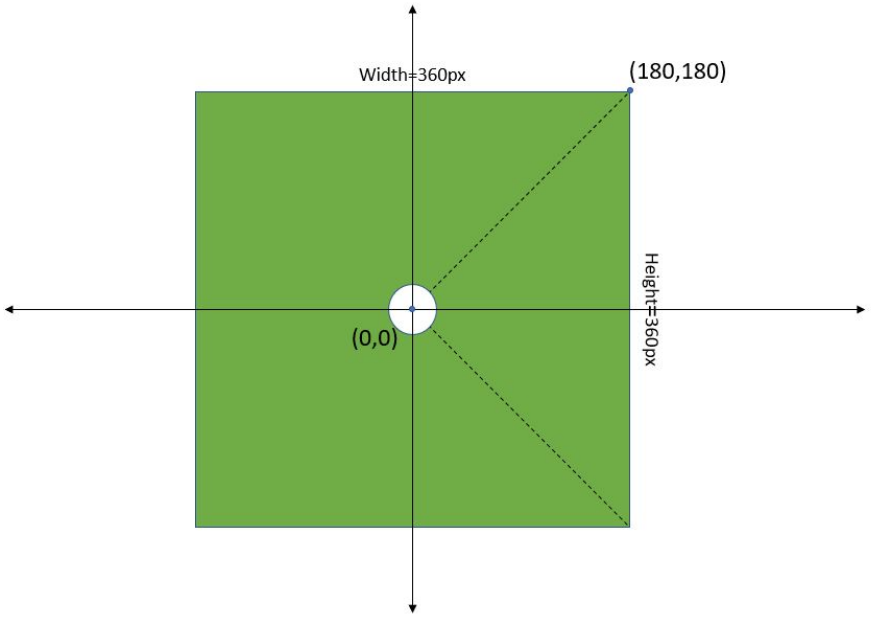
```
#Functions to move the object (goalkeeper)
def goalKeeper_up():
    y = goalKeeper.ycor() #to get the current y coordinate of the goalkeeper object
    y = y + goalKeeper_speed() #to increase current y coordinate by pixcels that the goalKeeper_speed() returns each time we move the goalkeeper up
    goalKeeper.sety(y) # to set the new y coordinate which is the old + specific px
    if (y>=145):
        y=145
        goalKeeper.sety(y)

def goalKeeper_down():
    y = goalKeeper.ycor() #to get the current y coordinate of the goalkeeper object
    y = y - goalKeeper_speed() #to decrease current y coordinate by pixcels that the goalKeeper_speed() returns each time we move the goalkeeper down
    goalKeeper.sety(y) # to set the new y coordinate which is the old - specific px
    if (y<=-140):
        y=-140
        goalKeeper.sety(y)
#End Functions of moving the object(goalkeeper)
window.onkeypress(goalKeeper_up,"w")
window.onkeypress(goalKeeper_down,"s")
```

Animation of The Ball Toward the Goalkeeper

We wanted the ball to be directed to the goal keeper in a specific range. This range is the right side of the window, we didn't want the ball to hit the top or the bottom of the window then to change its direction accordingly.



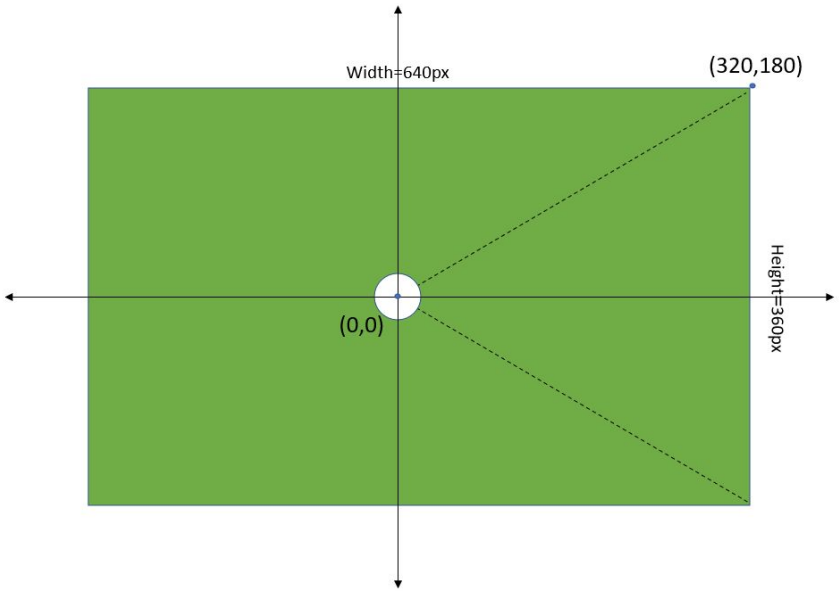
We found that if the window is squared and the ball is in the center of the window, we need to add one to the x coordinate of the ball and 1 to its y coordinate as well to get the corner point at most and without hitting the top of the window. As well as, we need to add 1 to x and -1 to y of the ball to reach the other corner point at least without hitting the bottom of the window.



$$\frac{x}{y} = \frac{180}{180} = 1$$

Means, to move from (0, 0) and get (180, 180), once we add 1 to x, we need to add 1 to y as well

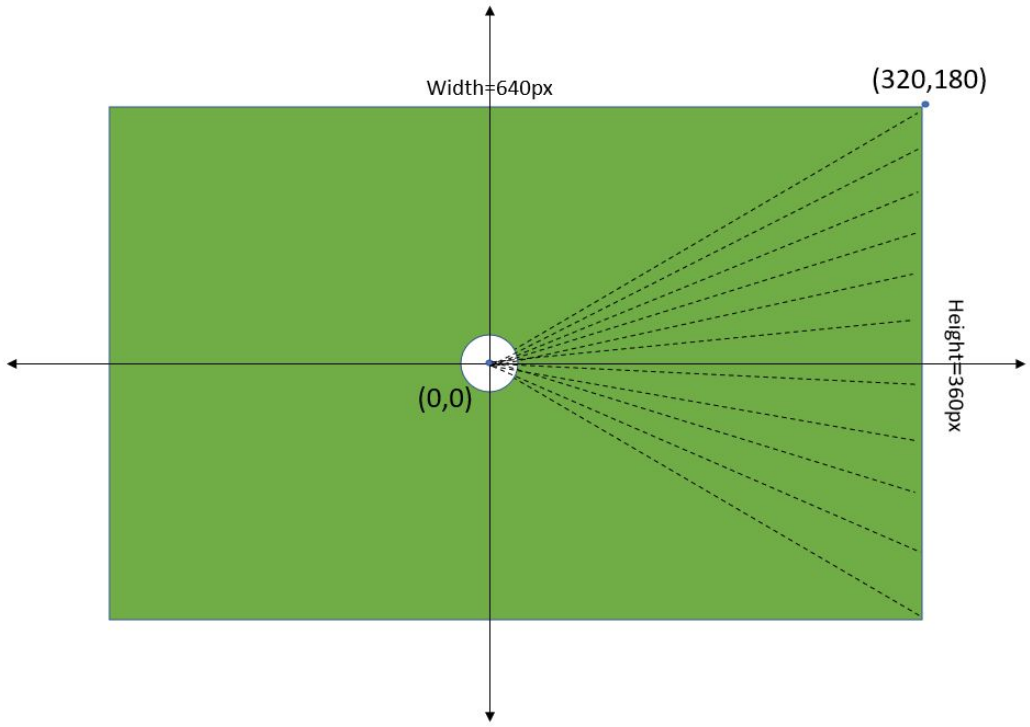
We followed the same way to find how much should be added to x and y coordinates of the ball to get the most and least points in the range if the window is rectangular and the ball is in the center.



$$\frac{x}{y} = \frac{320}{180} = 1.77$$

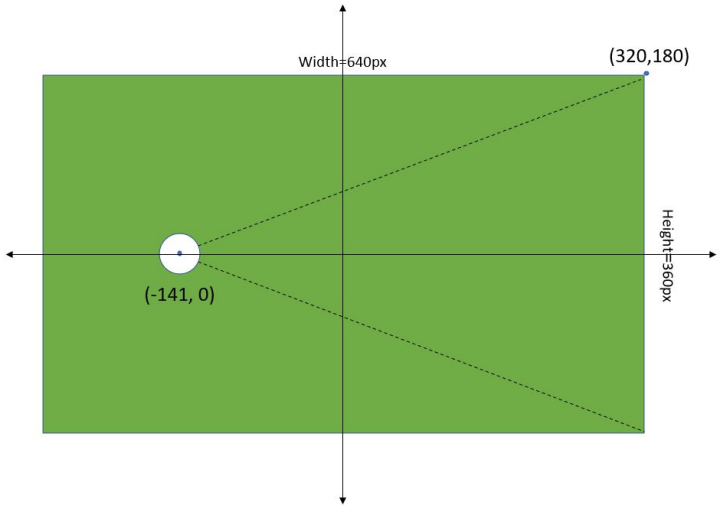
Means, to move from (0, 0) and get (320, 180), once we add 1.77 to x, we need to add 1 to y as well

We can add 1 as the most value to y coordinate of the ball if we add 1.77 to its x coordinate. We can add less that 1 to the y coordinate of the ball but no less than -1. If we add 1.77 to x as a fixed value and -1 to y of the ball at each time of the addition to x, the ball will get the other corner.



The value of x is fixed “one direction to the right”, and the value of y is in the range of -1 to 1 to change the direction of the ball vertically without going out of the range and hitting the top or bottom of the window before reaching the right side of the window.

However, the position of the ball in our game is not in the center of the window. So, we should get its position and do the same thing to know how much the fixed value of x and what is the range of the y value.

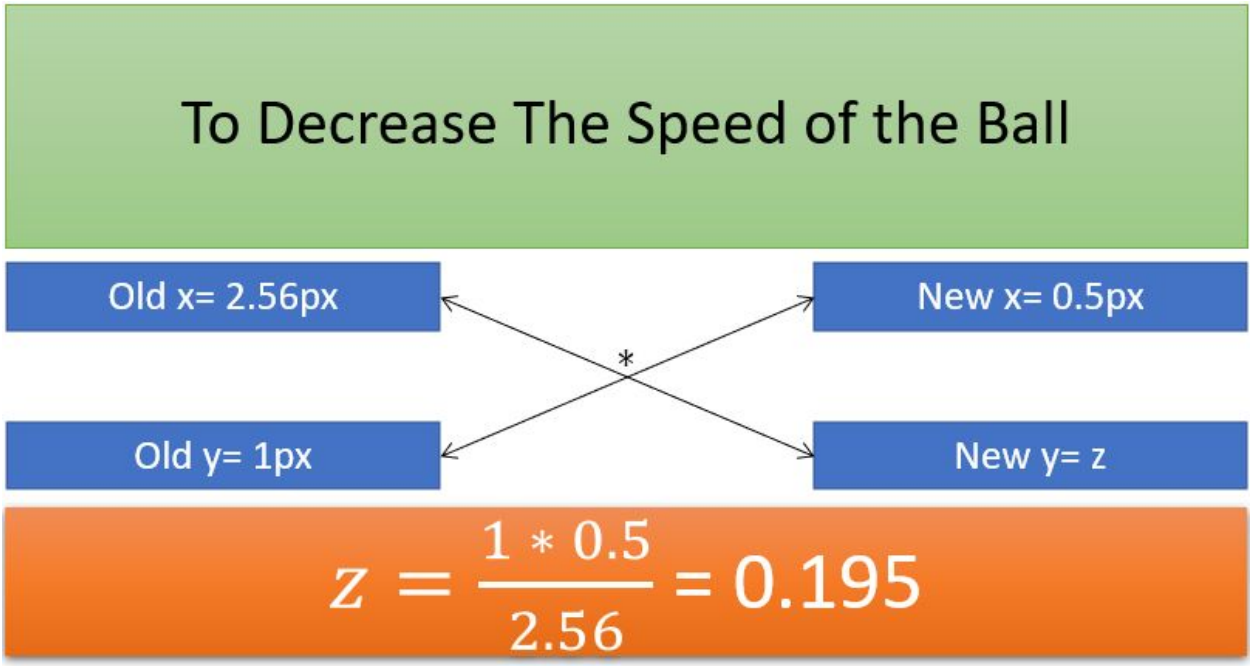


Means, to move from (-141, 0) and get (320, 180), the value of x will increase from -141 to 0, then 0 to 320. Means 461 and by dividing it by 180, the value will be 2.56

$$\frac{x}{y} = \frac{320 + 141}{180} = 2.56$$

For each addition to x by 2.56 px, we will add -1 to 1 to get the ($\mp$ 320, 180)

The value of 2.56 px is fixed and it determines the direction of the ball horizontally and because it is positive, it will be moved to the right side. The x coordinate of the ball will be increased by 2.56px for each loop. These pixels will determine the speed of the ball as well. So, we can decrease it or increase it to manipulate the speed of the ball. However, we should change the value of the y coordinate of the ball as well according to it to keep the range.



The result, any addition to x by 0.5px, the addition to the y should be in the range of -0.195px to 0.195px .

The initial speed of the ball in our game

When the game starts, the initial value to change the x coordinate of the ball is 0.3. We decided to give it this value in the level 1 of the game and this value will be added to x of the ball for each loop to let direct to the right. This value will be in px and it will be increased once the level of the game increases. The challenge will be increased as the speed of the ball increases. However, the y coordinate must be changed as well to keep the range that we want the ball to be directed according to. So, we used the equation we mentioned before to do that each time the level of the game increases as the x of the ball changes.

initial x = 0.3, initial y= 0.1171366594

$x = 2.561$ $x = 0.3$
y= 1 y= z

$$z = \frac{0.3}{2.561} = 0.1171366594$$

If the game level increases by one, 0.01 px will be added to the x coordinate of the ball to increase its value of going to the right for each loop. That means, the speed of the ball will be increased by 0.01 and the program will follow the equation again to increase the y coordinate of the ball accordingly to avoid losing the range.

After level up: x = 0.31, initial y= 0.1210412147

$x = 0.3$ $x = 0.31$
y= 0.1171366594 y= z

$$z = \frac{0.31 * 0.1171366594}{0.3} = 0.1210412147$$

Random Class for the y Coordinate of the Ball

We are going to animate the ball inside the playground to be pushed toward the goalkeeper who is on the right side. To let the ball move horizontally to the right, we need to change its x coordinate. The change will be positive to direct it to the right. So, for each loop, a specific value will be added to the x coordinate of the ball and this value will indicate the speed of the ball as well as the proportionality is directe.

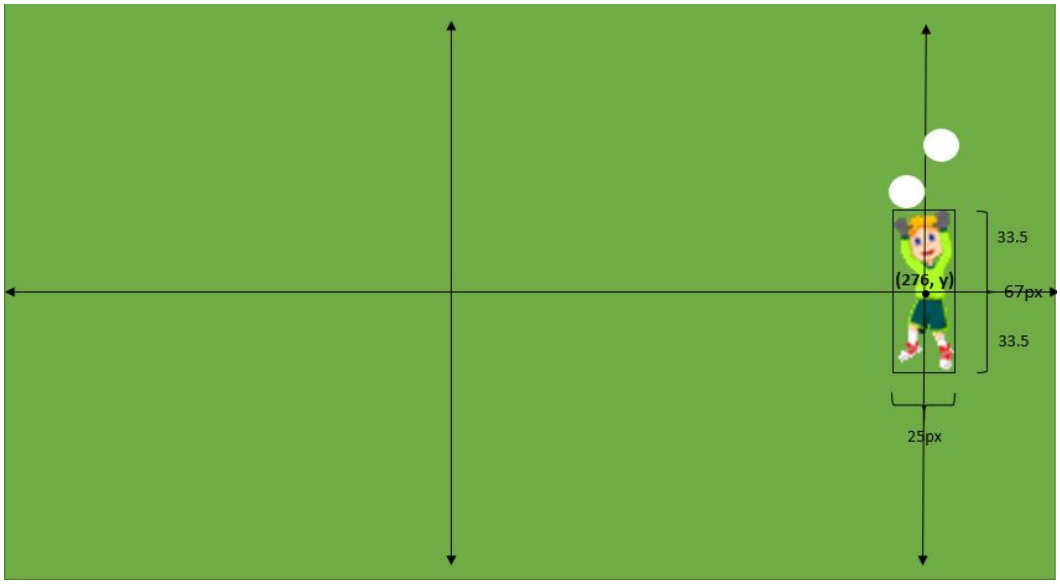
To make some manipulation to the direction of the ball to make it go to the right top or right bottom, we will choose a specific value to be added to the y coordinate of the ball in each loop. We used Random variable generators (uniform within range). The values of x will be read from the file to be added for each loop. The range will be read from the file as well to choose a value from the range by the random class to be added to the y for each loop.

If the ball hits the right side or the goal keeper catches it, it will be returned to its original place and another value will be chosen by the random class to be added to the y of the ball to change its direction. The value to be added to x of the ball will be the same till the level up.

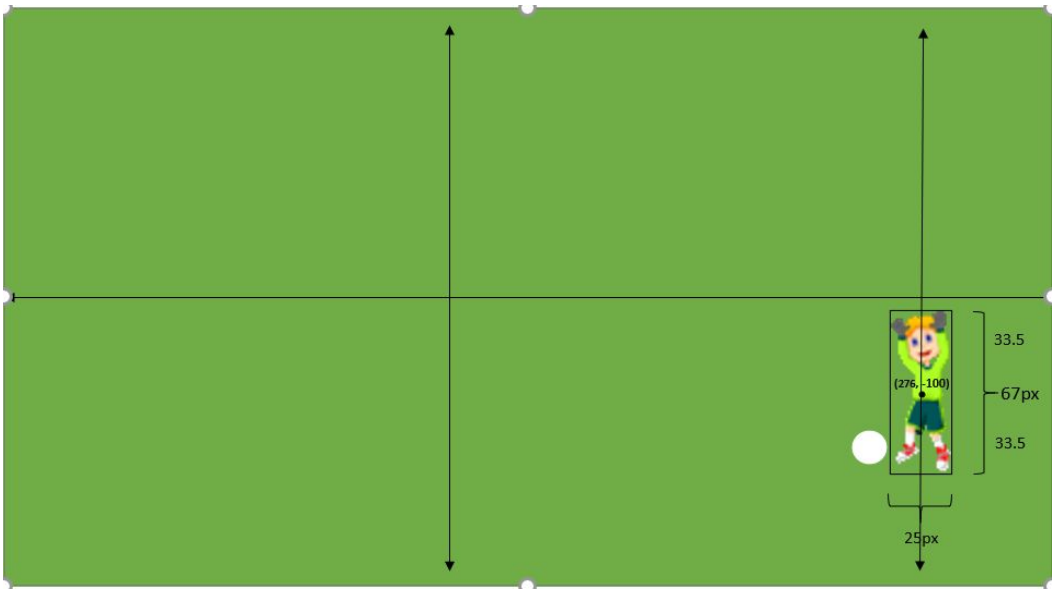
```
#by this, the ball will be pushed according to the height for the playgroung. Not going to hit the width of it
value_changing_x_ball=ball_speed_direction()[0] #this value will be added to x cor of the ball
#this random value in this range will be added to y cor of the ball
value_changing_y_ball=random.uniform(-ball_speed_direction()[1],ball_speed_direction()[1])
```

Game Conditions

1. If the goalkeeper doesn't catch the ball, means the ball xcor is more than the width of the window, the ball position will return to the original place. A value will be added to x cor of the ball as it is fixed and another random value to be added to ycor to change its value. incase of losing any ball, the progress will be 0 as the goalkeeper should catch the ball respectively till he reaches the number of the level to make it increased.
2. If the goalkeeper catch the ball:
    a. The x coordinate for the ball and the goalkeeper should be as this condition:
    `ball.xcor() > (276.0-(goalkeeper_width/2))  and  ball.xcor() < (276.0+(goalkeeper_width/2))`
    It is not only according to the center point of the goalkeeper body, but also to its width which is 25px. So, the x coordinate of the ball will be in the range of (276-12.5) and (276+12.5)



b. The y coordinate for the ball and the goalkeeper should be as this condition:
`ball.ycor() < (goalKeeper.ycor()+(goalkeeper_height/2))` and
`ball.ycor() > (goalKeeper.ycor()-(goalkeeper_height/2))`.
As long as the goalkeeper will be moving up and down, the y coordinate will be changing. So, once the x coordinate for the ball and the goalkeeper is as the previous condition, the y coordinate of the goalkeeper will be gotten at that time and it will be in the center point of his body. If his position is (276, -100), this will be the center point in his body and the ball will be catched only if its y coordinate is exactly -100px. However, we want the ball to be catched if it hits any point on his body, so we should consider his length. So, the y coordinate in this case should be equal to the range of (-100+33.5) to (-100-33.5).



Once he catches any ball, the progress will increase, and once the progress equals the current level, the level will increase by 1 and will be saved in the file and the progress will be 0 to start again to equal the next level. If the level equals 20, a Congratulations message will appear and the game will start from beginning with the initial values for the goalkeeper speed, ball speed, and the level.

Saving Data Using Files

Methods Used to Create and Read the Files:

- **file_create_then_write(path, value):** method to create the file if not exists, then to write in it.
- **file_read_line(path, line, exceptValue):** method to open the file and return the value in the first line or second line. If the file is not found, it will be created and the **exceptValue** will be inserted in it as initial value and to be returned as well.

We decided to create methods for reading and creating files because we found that we are dealing with the files several times to open and close it. So, by creating the methods, the code duplication will be decreased and to be called once needed. We include the try and except with the files to avoid the code from being terminated as the IOError may occur.

Methods uses the File Create and Read Functions

- **goalKeeper_speed():** To return the speed of the goalkeeper from the file. If the file does not exist, it will be created and 3.5px will be the initial value. When the player press w or s keys to call the goalKeeper_up() or goalKeeper_down() methods to change the position of him, this method 'goalKeeper_speed()' will be called there instead of having a specific value we write it ourselves, it will be gotten from the file.
- **ball_speed_direction():** To return the speed and direction 'x & y' of the ball from the file. If the file does not exist, it will be created and the values of 0.3px for x and 0.1171366594px as range for y initially will be inserted in.
- **get_level():** Method to get the level of the goalkeeper once the game starts to be shown later on the screen, if the file does not exist, it will be created and the initial level will be 1.
- **level_up(levelUp):** Function to increase the level of the goalkeeper after he catches a specific number of balls respectively according to the last level he got. To open the file and update the old value and put the new one coming by the param 'levelUp'. To increase the speed of the goalkeeper as well by updating it in the file "goalkeeperSpeedFile" by calling goalKeeper_speed() to get the last value in the file and increase it by 0.01. The equation we followed before to increase the speed of the ball and keep its range is implemented in this method.

Problems Solved

1. We observed that the program takes a long time with each running, we found out that it was because of the time consumed to create the pictures. The solution is to check if the image exists, means the game opened before and the playground is created, Therefore, no need to go through the code again. if an exception occurred, it will go to the except part to create it from scratch.

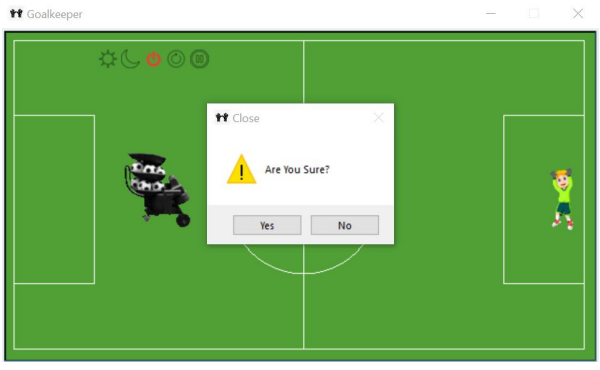
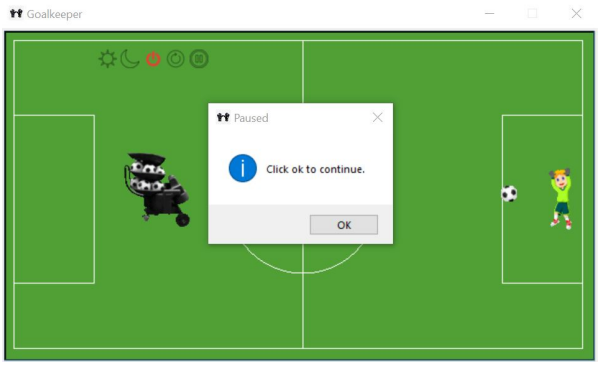
```
#to check if the image is existed, means the game opened before and the playground is created
#so, no need to go through the code again. if an exception accured, it will go to except part
#to create it from scratch
try:
    window.bgpic("images/backgroundDay.png")
except Exception as e:
    #Creating a black image in the background with the same size of the game window
    #it is colored with ch=3
    blackBackground=np.ones((gameScreenHeight,gameScreenWidth,3),dtype="uint8")



    #We are drawing shapes over the black image that we created

    #1- Green Rectangle is over the blackBackground
    playGround=cv2.rectangle(blackBackground,(0,0),(gameScreenWidth,gameScreenHeight),(80, 160, 54),-1)

    #2- Outer Border rectangle
    playGroundOuterBorder=cv2.rectangle(playGround,(10,10),(gameScreenWidth-20,gameScreenHeight-20),(255,255,255),1)

    #3- play Ground Middle Line
    playGroundOuterMiddleLine=cv2.line(playGround,(halfGameScreenWidth,10),(halfGameScreenWidth,gameScreenHeight-20),(255,255,255),1)

    #4- middle circle
    middleCircle=cv2.circle(playGround,(halfGameScreenWidth,halfGameScreenHeight),80,(255,255,255),1)

    #5- center point
    centerPoint=cv2.circle(playGround,(halfGameScreenWidth,halfGameScreenHeight),3,(255,255,255),-1)

    #6- penalty area 1
    penaltyArea1=cv2.rectangle(playGround,(gameScreenWidth-thirdOfhalfGameScreenWidth,quarterGameScreenHeight),(gameScreenWidth-20,gameScreenHeight-quarterGameScreenHeight),(25

    #7- penalty area 2
    penaltyArea2=cv2.rectangle(playGround,(10,quarterGameScreenHeight),(thirdOfhalfGameScreenWidth-10,gameScreenHeight-quarterGameScreenHeight),(255,255,255),1)

    #because imread method goes with the GBR, we convert the image to RGB
    #As well as, the background color of playGround was 54, 160, 80 with BGR and we edited it to 80, 160, 54 RGB above
    RGB_Image=cv2.cvtColor(blackBackground,cv2.COLOR_BGR2RGB)

    #saving the new background img that we created as normal img in images directory
    cv2.imwrite("images/backgroundDay.png",RGB_Image)

#to check if the image is existed, means the game opened before and the night image is created
#so, no need to go through the code again. if an exception accured, it will go to except part
#to create it
```

Creating the playground from scratch

```
#to check if the image is existed, means the game opened before and the night image is created
#so, no need to go through the code again. if an exception accured, it will go to except part
#to create it
try:
    window.bgpic("images/backgroundNight.png")
except Exception as e:
    #Creating the night mood image and save it with backgroundNight name in images directory
    RGB_Image2=cv2.imread("images/backgroundDay.png")
    for r in range(RGB_Image2.shape[0]):
        for c in range(RGB_Image2.shape[1]):
            for ch in range(RGB_Image2.shape[2]):
                RGB_Image2[r][c][ch]=RGB_Image2[r][c][ch]*0.45 #decrease 55% of the value of each color R, G, B
    cv2.imwrite("images/backgroundNight.png",RGB_Image2)
```

Creating the night mode vision and store it

2. We wanted the window to be opened with a button called “Play”. Once this button is clicked, the game will start (with all the animations at the beginning). So, we needed the whole code that we wrote till now to be inside a method called “start()”, including “the while True loop”. After that we faced a problem that the window didn't open like before. We have found that we need to include another while true loop outside the start() method or to include the mainloop() method related to the turtle model. We decided to choose the mainloop() method instead.
3. We faced another problem in which the objects animated slowly and there was shaking in the movements like the goalkeeper was blinking when hitting the top edge. After searching, we found that the window is updating automatically and we need to call a method called “tracer” to Stop the window from updating automatically.
4. We faced another problem when we opened the window, the code terminated suddenly. This didn't always happen. After some research on the turtle library We have found a variable called TurtleScreen._RUNNING. If this variable is set to “True”, a turtle window opens and if not, the turtle.Terminator error will be occured. TurtleScreen._RUNNING=True .
5. Confirmation message box will be shown when closing or restarting the game by clicking on the “close” or “restart” buttons. By using the messagebox from tkinter package. “askquestion() and showinfo()” are called for that job. For closing the window, we used the bye() method in turtle model.

Confirmation Message	Information Message
	

Results

The result of this project will be a game that the user can control. The window will be opened after running which contains a goalkeeper and a ball launcher. The player will try to block as many balls as possible to win, otherwise the loss will be inevitable. Modes, effects, animations, and etc. will be in the game when starting and closing

Conclusion

In conclusion, using python programming language for graphics is very effective and easy. The programmer will be able to use the models like turtle, tkinter, etc. to start creating the graphics in a very effective way. Turtle graphics is a popular way for creating objects and animating them easily, the output of the created application is a responsive game that has effects after starting depending on the generation codes that generates the game playground, objects will be animated and moved to the playground automatically after the user press start, then the game will starting, during the game user can interact with the game, cached balls calculated and user level changes according to the amount of cached balls. Also there are buttons to control the game like switching between day and night modes, switching off the game or restart the game. After closing the game a paint screen will appear to paint animated bye words on the screen.

References

https://stackoverflow.com/
https://www.w3schools.com/
https://docs.python.org/3/library/turtle.html
https://www.tutorialspoint.com/